质数

2019年2月25日 15:49

- 质数的判定

一. 定义

1. 质数：除1和自身外不能被任何自然数整除的正整数
2. 合数：不是质数的正整数
3. N足够大时不超过N的质数约N/lnN个，即lnN个数中约有一个质数

二. 试除法

1. 反证法知遍历[2,sqrt(n)]就能判定
2.
```
bool is_prime(int n){
    for__(i,2,sqrt(n)) if(n%i==0) return false;
return true;}
```

三. Miller-Rabin测试（HDU2138）

1. 二次探测：0<x<p,p是质数 x^2 \equiv 1(\mod p) \Leftrightarrow x=1或x=p-1
2. 费马探测：0<a<p,p是质数 \Rightarrow a^{p-1} \equiv 1(\mod p)
3. 判错概率<4^{-S}(S为测试次数)，时间复杂度O（S*logn）

```
inline ll qpow(ll a,ll b,ll P){  //a^b%P
    ll ans=1;
    for(; b; b>>=1,a=a*a%P)
        if(b&1) ans=ans*a%P;
return ans;}

inline bool MR(int tc,ll n){     //用随机测试数据tc测n，返回n是质数?1:0
    ll u=n-1, t=0;
    while(!(u&1)) ++t, u>>=1; //把n-1表示为u*2^t，循环到u为奇数为止
    ll x=qpow(tc,u,n);
    if(x==1) return 0;
    for(int i=1; i<=t; ++i,x=x*x%n)
        if(x!=n-1&&x!=1&&x*x%n==1) return 1;
    return x!=1;
}

mt19937_64 rnd(time(0));  //c++11的std的魔法，下文可用rnd()生成[0,2^64)

bool isPrime(ll n,int S=10){     //对n测试S次，返回n是质数?1:0
    if(n==2) return 1;
    if(n<2 || !(n&1)) return 0;
//    srand(time(0));
    while(S--)
//          if(MR((ll)rand()*rand()*rand()%(n-1)+1,n)) return 0;
        if(MR(rnd()%(n-1)+1,n)) return 0;
    return 1;
}
```

- 质数的筛选

四. Eratosthens筛法

1. 基本思想：i>=2时，整数x的整数i倍都不是质数
2. 优化：小于x方的倍数都被小于x的数筛过
3. 时间复杂度O（NloglogN）

4.
```
void primes(int N){
     memset(npr,0,sizeof(npr));
     for_(i,2,N){
          if(npr[i]) continue;
          cout<<i<<"是质数"<<endl;
          for_(j,i,N/i) npr[i*j]=true;}}
```

五. 欧拉线性筛

1. 因为每个数只会被最小质因子v[i]筛一次，因此复杂度O（n）

```
int prm[MN],tot;    //prime质数集合及其大小
int v[MN];          //v[i] = i的最小质因子
void init_prm(int N){     //打表[1,N]的prm
     for(int i=2; i<=N; ++i){
          if(!v[i]) v[i] = i, prm[tot++] = i;
          for(int j=0; j<tot; ++j){
               int p = prm[j];
               if(i*p > N || p>v[i]) break;
               v[i*p] = p;}}}
```

2. 不计最小质因子的筛

```
int prm[MN],tot;    //prime质数集合及其大小
bitset<MN> v;       //v[i] = i不是质数吗?
void init_prm(int N){     //打表[1,N]的prm
     for(int i=2; i<=N; ++i){
          if(!v[i]) v[i] = i, prm[tot++] = i;
          for(int j=0; j<tot; ++j){
               int p = prm[j];
               if(i*p > N) break;
               v[i*p] = 1;
               if(i%p == 0) break;}}}
```

-
- 例题

六. 求区间[l,r]中恰能分解为两个质因数的积的合数，及其两个质因数（同济）

1. 范围较大，可以先给i映射到d[i-l]上
2. r较大，无法给[2,r]所有质数打表，但较小质数一定<=sqrt(r)，可以给较小质数打表[2,sqrt(r)]
3. 对d[i]遍历质数表的j，用while(d[i]%j==0) d[i]/=j;不断做除法，分解剩下的商因为也<=r，也在sqrt(r)前无法分解，所以它也是质数
4. 这样就能给d[i]质因数分解了，把分解的结果保存在vector中
5. 但是如果对暴力遍历d[i]和表内所以j还是比较慢，应该类似筛质数的用d[i]+=j来跳过肯定不能整除的d[i]
6. 这样的筛法，时间复杂度和调和级数有关，因而也是O（NlogN）
7. 注意这个筛法的起点，是向上取整(l+j-1)/j*j，另外int范围也要注意
8. 合数的倍数一定也是其质因数的倍数，因此不筛出质数表也不会出错，在十万以内差距也不大

```
scanf("%lld%lld",&l,&r);
for(ll i=l; i<=r; ++i) d[i-l]=i;    //映射到存得下的下标
int rr=sqrt(r);
for__(i,2,rr)
     if(!nprm[i]){
          prm.push_back(i);
```

```
          ll j=(ll)i*i;
          while(j<=rr)
               nprm[j]=1,
               j+=i;}
int J=prm.size();

for_(k,0,J){
     int j=prm[k];
     for(int i=(l+j-1)/j*j-l; i+l<=r; i+=j)
          while(d[i]>1 && d[i]% j== 0)
               ans[i].push_back(j),
               d[i]/=j;}
rof__(i,r-l,0){
     if(d[i]>1)
          ans[i].push_back(d[i]);
     if(ans[i].size()==2)
          ++ans0,
          sort(ans[i].begin(),ans[i].end());}
printf("%d\n",ans0);
for__(i,0,r-l)
     if(ans[i].size()==2)
          printf("%lld %lld %lld\n",l+i,ans[i][0],ans[i][1]);}
```